Package ‘sfheaders’

March 19, 2020

Type  Package
Title  Converts Between R Objects and Simple Feature Objects
Date  2020-03-19
Version  0.2.1
Description  Converts between R and Simple Feature ‘sf’ objects, without depending on the Simple Feature library. Conversion functions are available at both the R level, and through ‘Rcpp’.
License  GPL-3
URL  https://dcooley.github.io/sfheaders/
BugReports  https://github.com/dcooley/sfheaders/issues
Encoding  UTF-8
LazyData  true
RoxygenNote  7.0.2
SystemRequirements  C++11
LinkingTo  Rcpp
Imports  Rcpp
Suggests  covr, knitr, testthat
NeedsCompilation  yes
Author  David Cooley [aut, cre]
Maintainer  David Cooley <david.cooley.au@gmail.com>
Repository  CRAN
Date/Publication  2020-03-19 18:30:06 UTC

R topics documented:

  sfc_cast ......................................................... 2
  sfc_linestring .................................................. 3
  sfc_multilinestring ........................................... 4
  sfc_multipoint .................................................. 6
**Description**

convert the input sfc to a different geometry

**Usage**

`sfc_cast(sfc, to, close = TRUE)`

**Arguments**

- `sfc` : geometry object to convert to a different geometry
- `to` : the geometry to convert to.
- `close` : logical indicating if polygons should be closed
Examples

```r
df <- data.frame(
  id1 = c(1,1,1,1,1,1,1,1,2,2,2,2),
  id2 = c(1,1,1,2,2,2,2,1,1,1,1),
  x = c(0,0,1,1,1,1,2,2,3,4,4,3),
  y = c(0,1,1,0,2,2,1,3,3,4,4,4)
)
pt <- sfc_point(obj = df, x = "x", y = "y", z = "id1")
mpt <- sfc_multipoint(obj = df, x = "x", y = "y", multipoint_id = "id1")
ls <- sfc_linestring(obj = df, x = "x", y = "y", linestring_id = "id1")
mls <- sfc_multilinestring(obj = df, x = "x", y = "y", multilinestring_id = "id1")
p <- sfc_polygon(
  obj = df,
  x = "x",
  y = "y",
  polygon_id = "id1",
  linestring_id = "id2",
  close = FALSE
)
mp <- sfc_multipolygon(
  obj = df,
  x = "x",
  y = "y",
  multipolygon_id = "id1",
  linestring_id = "id2",
  close = FALSE
)
sfc_cast( pt, "LINESTRING" )
sfc_cast( mpt, "POLYGON" )
sfc_cast( ls, "POINT" )
sfc_cast( mls, "MULTIPOLYGON" )
sfc_cast( p, "POINT" )
sfc_cast( mp, "LINESTRING" )
```

Description

constructs sfc of MULTIPOINT objects

Usage

```r
sfc_linestring(
  obj = NULL,
  x = NULL,
  y = NULL,
  linestring_id = NULL,
  multipoint_id = NULL,
  multilinestring_id = NULL,
  close = FALSE
)```
sfc_multilinestring

```r
y = NULL,
z = NULL,
m = NULL,
linestring_id = NULL
```

Arguments

- `obj`: sorted matrix or data.frame
- `x`: x geometry column
- `y`: y geometry column
- `z`: z geometry column
- `m`: m geometry column
- `linestring_id`: column of ids for linestrings

Value

- `sfc` object of LINESTRING geometries

Notes

- sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.
- The data.frame and matrices you send into the sfheader functions must be ordered.

Examples

```r
x <- matrix( c(1:4), ncol = 2 )
sfc_linestring( x )

x <- data.frame( id = 1:2, x = 1:2, y = 2:1 )
sfc_linestring( x )
sfc_linestring( x, x = "x", y = "y" )
sfc_linestring( x, x = "y", y = "x" )
sfc_linestring( x, linestring_id = "id", x = "x", y = "y" )
```

---

sfc_multilinestring  sfc MULTILINESTRING

Description

- constructs an sfc of MULTILINESTRING objects
sfc_multilinestring

Usage

sfc_multilinestring(
  obj = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  multilinestring_id = NULL,
  linestring_id = NULL
)

Arguments

obj            sorted matrix or data.frame
x              x geometry column
y              y geometry column
z              z geometry column
m              m geometry column
multilinestring_id
               column of ids for multilinestrings
linestring_id  column of ids for linestrings (within multilinestrings)

Value

sfc object of MULTILINESTRING geometries

notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

Examples

m <- matrix(c(0,0,0,1,1), ncol = 3 )
sfc_multilinestring( m )

m <- matrix(c(0,0,0,0,1,1,1,1,2,2,1,2,3), ncol = 3, byrow = TRUE)
sfc_multilinestring( obj = m )
sfc_multilinestring( obj = m, multilinestring_id = 1 )
sfc_multilinestring( obj = m, linestring_id = 1 )
sfc_multilinestring( obj = m, linestring_id = 1, multilinestring_id = 1 )

sfc_multilinestring( obj = m, x = 2, y = 3 )
sfc_multilinestring( obj = m, x = 1, y = 2, z = 3 )
sfc_multilinestring( obj = m, x = 2, y = 3, linestring_id = 1, multilinestring_id = 1 )
df <- data.frame(
  ml_id = c(1,1,1,1,1,1,1,1,2,2,2,2,2),
  l_id = c(1,1,1,2,2,3,3,3,1,1,1,2,2),
  x = rnorm(13),
  y = rnorm(13),
  z = rnorm(13),
  m = rnorm(13)
)

sfc_multilinestring( obj = df, x = "x", y = "y")
sfc_multilinestring( obj = df, x = "x", y = "y", z = "z")
sfc_multilinestring( obj = df, x = "x", y = "y", z = "z", m = "m")

sfc_multilinestring( obj = df, x = 2, y = 3)
sfc_multilinestring( obj = df, x = 2, y = 3, z = 4)
sfc_multilinestring( obj = df, x = 2, y = 3, z = 4, m = 5)

sfc_multilinestring( obj = df, multilinestring_id = "ml_id", linestring_id = "l_id" )
sfc_multilinestring( obj = df, multilinestring_id = 1, linestring_id = 2 )

---

sfc_multipoint

**sfc MULTIPOINT**

**Description**

constructs sfc of MULTIPOINT objects

**Usage**

sfc_multipoint(
  obj,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  multipoint_id = NULL
)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>sorted matrix or data.frame</td>
</tr>
<tr>
<td>x</td>
<td>x geometry column</td>
</tr>
<tr>
<td>y</td>
<td>y geometry column</td>
</tr>
<tr>
<td>z</td>
<td>z geometry column</td>
</tr>
</tbody>
</table>
**sfc_multipolygon**

m m geometry column
multipoint_id column of ids for multipoints

**Value**

sfc object of MULTIPOINT geometries

**notes**

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

**Examples**

```r
x <- matrix( c(1:4), ncol = 2 )
sfc_multipoint( x )

x <- data.frame( id = 1:2, x = 1:2, y = 2:1 )
sfc_multipoint( x )
sfc_multipoint( x, x = "x", y = "y" )
sfc_multipoint( x, x = "y", y = "x" )
sfc_multipoint( x, multipoint_id = "id", x = "x", y = "y")
```

---

**sfc_multipolygon**

**sfc MULTIPOLYGON**

**Description**

constructs an sfc of MULTIPOLYGON objects

**Usage**

```r
sfc_multipolygon(
obj = NULL,
x = NULL,
y = NULL,
z = NULL,
m = NULL,
multipolygon_id = NULL,
polygon_id = NULL,
linestring_id = NULL,
close = TRUE
)
```
Arguments

obj sorted matrix or data.frame
x x geometry column
y y geometry column
z z geometry column
m m geometry column
multipolygon_id column of ids for multipolygons
polygon_id column of ids for polygons
linestring_id column of ids for lines (within polygons)
close logical indicating whether polygons should be closed. If TRUE, all polygons will be checked and force closed if possible

Value

sfc object of MULTIPOLYGON geometries

notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

Examples

m <- matrix(c(0,0,0,1,0,0,1,1,0,0,1,0,0,0), ncol = 3, byrow = TRUE )
sfc_multipolygon(m)

df <- data.frame(
  id = c(1,1,1,1),
  x = c(0,0,1,1),
  y = c(0,1,1,0)
)
sfc_multipolygon(df, x = "x", y = "y")

df <- data.frame(
  id = c(1,1,1,1,2,2,2,2),
  x = c(0,0,1,0,1,2,2,1),
  y = c(0,1,0,1,2,2,1,1)
)
sfc_multipolygon(df, multipolygon_id = "id", polygon_id = "id", linestring_id = "id")

df <- data.frame(
  id1 = c(1,1,1,1,1,1,1,1),
  id2 = c(1,1,1,1,2,2,2,2)
sfc_multipolygon( df, multipolygon_id = "id1", polygon_id = "id2"

df <- data.frame(
  id1 = c(1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2),
  id2 = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
  x = c(0,0,0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2),
  y = c(0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
)

sfc_multipolygon( df, multipolygon_id = "id1", polygon_id = "id2"

sfc_multipolygon( df, polygon_id = "id1", linestring_id = "id2"

sfc_multipolygon( df, x = "x", y = "y", polygon_id = "id1"

sfc_multipolygon( df, x = "x", y = "y", linestring_id = "id2"

sfc_multipolygon( df, x = "x", y = "y", polygon_id = "id1"

sfc_multipolygon( df, x = "x", y = "y", linestring_id = "id2"

df <- data.frame(
  id1 = c("a","a","a","a","a","b","b","b","b","b","b","b","b","b","b","b","b","b"),
  id2 = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
  x = c(0,0,1,1,0,1,1,2,2,1,2,2,1,2,2,1,2,2),
  y = c(0,1,1,0,0,1,2,2,1,1,1,1,1,1,1,1,1,1)
)

sfc_multipolygon( df, x = "x", y = "y", polygon_id = "id1"

sfc_multipolygon( df, x = "x", y = "y", polygon_id = "id1"

sfc_multipolygon( df, x = "x", y = "y", linestring_id = "id2"

sfc_multipolygon( df, x = "x", y = "y", linestring_id = "id2"

sfc_multipolygon( df, x = "x", y = "y", polygon_id = "id1"

sfc_multipolygon( df, x = "x", y = "y", linestring_id = "id2"

---

**sfc_point**

**sfc POINT**

**Description**

constructs sfc of POINT objects

**Usage**

sfc_point(obj, x = NULL, y = NULL, z = NULL, m = NULL)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>sorted vector, matrix or data.frame</td>
</tr>
<tr>
<td>x</td>
<td>x geometry column</td>
</tr>
<tr>
<td>y</td>
<td>y geometry column</td>
</tr>
<tr>
<td>z</td>
<td>z geometry column</td>
</tr>
<tr>
<td>m</td>
<td>m geometry column</td>
</tr>
</tbody>
</table>

Value

sfc object of POINT geometries

Notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

Examples

```
x <- c(1:3)
sfc_point( x )

x <- matrix( c(1:10) , ncol = 2 )
sfc_point( x )

x <- setNames( as.data.frame( x ), c("x","y") )
sfc_point( x )
sfc_point( obj = x, x = "x", y = "y" )
sfc_point( obj = x, x = "y", y = "x" )
```

sfc_polygon

Description

constructs an sfc of POLYGON objects

Usage

```
sfc_polygon(  
    obj = NULL,  
    x = NULL,  
    y = NULL,  
    z = NULL,  
)```
m = NULL,
polygon_id = NULL,
linestring_id = NULL,
close = TRUE
)

Arguments

obj    sorted matrix or data.frame
x      x geometry column
y      y geometry column
z      z geometry column
m      m geometry column
polygon_id    column of ids for polygons
linestring_id    column of ids for lines (within polygons)
close    logical indicating whether polygons should be closed. If TRUE, all polygons will
         be checked and force closed if possible

Value

sfc object of POLYGON geometries

notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

Examples

m <- matrix(c(0,0,0,1,1), ncol = 2)
sfc_polygon( m )

m <- matrix(c(0,0,0,0,1,1,1,1,2,2,1,2,3,1,3,2), ncol = 3, byrow = TRUE)
sfc_polygon( obj = m )
sfc_polygon( obj = m, polygon_id = 1 )
sfc_polygon( obj = m, linestring_id = 1 )
sfc_polygon( obj = m, linestring_id = 1, polygon_id = 1 )
sfc_polygon( obj = m, x = 2, y = 3 )
sfc_polygon( obj = m, x = 1, y = 2, z = 3 )
sfc_polygon( obj = m, x = 2, y = 3, linestring_id = 1, polygon_id = 1 )

df <- data.frame(  
  ml_id = c(1,1,1,1,1,1,1,1,2,2,2,2,2,2,2),
  l_id = c(1,1,1,2,2,2,3,3,3,1,1,2,2,2,2)  
)
```r
sfc_to_df

sfc_polygon( obj = df, x = "x", y = "y"
)  
sfc_polygon( obj = df, x = "x", y = "y", z = "z"
)  
sfc_polygon( obj = df, x = "x", y = "y", z = "z", m = "m"
)
```

```r
sfc_polygon( obj = df, x = 2, y = 3)
sfc_polygon( obj = df, x = 2, y = 3, z = 4)
sfc_polygon( obj = df, x = 2, y = 3, z = 4, m = 5)
```

```r
sfc_polygon( obj = df, polygon_id = "ml_id", linestring_id = "l_id"
)  
sfc_polygon( obj = df, polygon_id = 1, linestring_id = 2
)
```

**Description**

Converts an sfc object to a data.frame

**Usage**

```r
sfc_to_df(sfc)
```

**Arguments**

```r
sfc  
sfc object
```

**Examples**

```r
x <- matrix( c(1:16), ncol = 2 )
sfc <- sfc_linestring( x )
df <- sfc_to_df( sfc )
```

```r
df <- data.frame(
  ml_id = c(1,1,1,1,1,1,1,1,2,2,2,2,2,2,2),
  l_id = c(1,1,1,1,1,1,1,1,2,2,2,2,2,2,2)
)
```
**sfg_linestring**

```r
sfc <- sfc_multilinestring( obj = df, multilinestring_id = "ml_id", linestring_id = "l_id" )
df <- sfc_to_df( sfc )
```

---

**sfg_linestring**    **sfg linestring**

---

**Description**

constructs sfg LINESTRING object

**Usage**

```r
sfg_linestring(obj, x = NULL, y = NULL, z = NULL, m = NULL)
```

**Arguments**

- **obj**: matrix or data.frame
- **x**: x geometry column
- **y**: y geometry column
- **z**: z geometry column
- **m**: m geometry column

**Value**

sfg object of LINESTRING geometry

**Examples**

```r
sfg_linestring( matrix( 1:24, ncol = 2 ) )
sfg_linestring( matrix( 1:24, ncol = 3 ) )
sfg_linestring( matrix( 1:24, ncol = 4 ) )
sfg_linestring( matrix( 1:24, ncol = 4 ), x = 3, y = 2, z = 3)
sfg_linestring( data.frame( x = 1:10, y = 11:20 ) )
sfg_linestring( data.frame( x = 1:10, y = 11:20, z = 21:30 ) )
sfg_linestring( data.frame( x = 1:10, y = 11:20, z = 21:30 ), x = "x", y = "z" )
```
Description

constructs sfg MULTILINESTRING object

Usage

sfg_multilinestring(
  obj,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  linestring_id = NULL
)

Arguments

obj     matrix or data.frame
x       x geometry column
y       y geometry column
z       z geometry column
m       m geometry column
linestring_id column of ids for lines

Value

sfg object of MULTILINESTRING geometry

Examples

sfg_multilinestring( matrix( 1:24, ncol = 2 ) )
sfg_multilinestring( matrix( 1:24, ncol = 3 ) )
sfg_multilinestring( matrix( 1:24, ncol = 4 ) )

## different lines
m <- cbind( matrix( 1:24, ncol = 2 ), c(rep(1, 6), rep(2, 6) ) )
sfg_multilinestring( obj = m, x = 1, y = 2, linestring_id = 3 )

## just specifying linestring_id will use all others as the geometries
sfg_multilinestring( obj = m, linestring_id = 3 )

df <- data.frame( x = 1:12, y = 1:12, z = 13:24, id = c(rep(1,6), rep(2,6)))
sfg_multilinestring( df, x = "x", y = "y" )
sfg_multipoint

sfg_multilinestring( df, x = "x", y = "y", linestring_id = "id" )

sfg_multilinestring( df, linestring_id = "id" )

sfg_multipoint

Description

constructs sfg MULTIPOINT object

Usage

sfg_multipoint(obj, x = NULL, y = NULL, z = NULL, m = NULL)

Arguments

obj matrix or data.frame
x x geometry column
y y geometry column
z z geometry column
m m geometry column

Value

sfg object of MULTIPOINT geometry

Examples

sfg_multipoint( 1:2 )
sfg_multipoint( 1:3 )
sfg_multipoint( 1:4 )

sfg_multipoint( matrix( 1:3, ncol = 3 ) )
sfg_multipoint( data.frame( x = 1, y = 2, z = 3 ) )

sfg_multipoint( matrix( 1:4, ncol = 2 ) )
sfg_multipoint( matrix( 1:24, ncol = 2, byrow = TRUE ) )
sfg_multipoint( matrix( 1:24, ncol = 3, byrow = TRUE ) )
sfg_multipoint( matrix( 1:24, ncol = 4, byrow = TRUE ) )
sfg_multipoint( data.frame( x = 1:5, y = 1:5 ) )

## using columns
sfg_multipoint( matrix( 1:24, ncol = 4, byrow = TRUE ), x = 1, y = 2 )
sfg_multipoint( matrix( 1:24, ncol = 4, byrow = TRUE ), x = 1, y = 2, z = 3 )
sfg_multipoint( matrix( 1:24, ncol = 4, byrow = TRUE ), x = 3, y = 4 )

df <- data.frame( x = 1:5, y = 1:5, z = 11:15, m = 11:15 )
sfg_multipoint( df, x = "x", y = "y" )
sfg_multipoint( df, x = "x", y = "y", z = "z" )
sfg_multipoint( df, x = "x", y = "y", z = "z", m = "m" )

---

**Description**

constructs sfg MULTIPOLYGON object

**Usage**

```r
sfg_multipolygon(
  obj, 
  x = NULL, 
  y = NULL, 
  z = NULL, 
  m = NULL, 
  polygon_id = NULL, 
  linestring_id = NULL, 
  close = TRUE
)
```

**Arguments**

- `obj` matrix or data.frame
- `x` x geometry column
- `y` y geometry column
- `z` z geometry column
- `m` m geometry column
- `polygon_id` column of ids for polygons (within the multipolygon)
- `linestring_id` column of ids for lines (within polygons)
- `close` logical indicating whether polygons should be closed. If TRUE, all polygons will be checked and force closed if possible

**Value**

sfg object of MULTIPOLYGON geometry
Examples

```r
df <- data.frame(
  polygon_id = c(rep(1, 5), rep(2, 10)),
  line_id = c(rep(1, 10), rep(2, 5)),
  x = c(0,0,1,1,0,2,2,5,5,2,3,3,4,4,3),
  y = c(0,1,1,0,0,2,5,5,2,2,3,4,4,3),
  z = c(1),
  m = c(1)
)

m <- as.matrix(df)

sfg_multipolygon(df[, c("x","y")])

sfg_multipolygon(df, x = "x", y = "y", polygon_id = "polygon_id", linestring_id = "line_id")

sfg_multipolygon(df, x = "x", y = "y", z = "z", polygon_id = "polygon_id", linestring_id = "line_id")

sfg_multipolygon(df, x = "x", y = "y", z = "z", m = "m", polygon_id = "polygon_id", linestring_id = "line_id")

sfg_multipolygon(m[, c("x","y")])

sfg_multipolygon(m, x = "x", y = "y", polygon_id = "polygon_id", linestring_id = "line_id")

sfg_multipolygon(m, x = "x", y = "y", z = "z", polygon_id = "polygon_id", linestring_id = "line_id")

sfg_multipolygon(m, x = "x", y = "y", z = "z", m = "m", polygon_id = "polygon_id", linestring_id = "line_id")
```

sfg_point

Description

constructs sfg POINT object

Usage

```r
sfg_point(obj, x = NULL, y = NULL, z = NULL, m = NULL)
```
Arguments

obj matrix or data.frame
x x geometry column
y y geometry column
z z geometry column
m m geometry column

Value

sfg object of POINT geometry

Examples

sfg_point(1:2)
sfg_point(1:3)
sfg_point(1:4)
sfg_point(matrix(1:3, ncol = 3))
sfg_point(data.frame(x = 1, y = 2, z = 3))
sfg_point(data.frame(x = 1, y = 2, z = 3), x = "x", y = "y")
sfg_point(data.frame(x = 1, y = 2, z = 3), x = 1, y = 3)

Description

constructs sfg POLYGON object

Usage

sfg_polygon(
    obj,
    x = NULL,
    y = NULL,
    z = NULL,
    m = NULL,
    linestring_id = NULL,
    close = TRUE
)
Arguments

- **obj**: matrix or data.frame
- **x**: x geometry column
- **y**: y geometry column
- **z**: z geometry column
- **m**: m geometry column
- **linestring_id**: column of ids for lines (within polygons)
- **close**: logical indicating whether polygons should be closed. If TRUE, all polygons will be checked and force closed if possible

Value

sfg object of POLYGON geometry

Examples

```r
sfg_polygon( matrix( 1:24, ncol = 2 ) )
sfg_polygon( matrix( 1:24, ncol = 3 ) )
sfg_polygon( matrix( 1:24, ncol = 4 ) )
```

```r
## different lines
m <- cbind( matrix( 1:24, ncol = 2 ), c(rep(1, 6), rep(2, 6) ) )
sfg_polygon( obj = m, x = 1, y = 2, linestring_id = 3 )
```

```r
## just specifying linestring_id will use all others as the geometries
sfg_polygon( obj = m, linestring_id = 3 )
```

```r
df <- data.frame( x = 1:12, y = 1:12, z = 13:24, id = c(rep(1,6), rep(2,6)))
sfg_polygon( df, x = "x", y = "y" )
sfg_polygon( df, x = "x", y = "y", linestring_id = "id" )
```

sfg_polygon( df, linestring_id = "id" )

---

**sfg_to_df**

**sfg to df**

Description

Converts an sfg object to a data.frame

Usage

sfg_to_df(sfg)
Arguments

sfg            sfg object

Examples

sfg <- sfg_point( obj = c(1,2) )
df <- sfg_to_df( sfg )

m <- cbind( matrix( 1:24, ncol = 2 ), c(rep(1, 6), rep(2, 6) ) )
sfg <- sfg_polygon( obj = m, x = 1, y = 2, linestring_id = 3 )
df <- sfg_to_df( sfg )

sf_bbox

Description

Calculates the bounding box of coordinates. This does not read the "bbox" attribute, it re-calculates the bounding box from the geometry coordinates

Usage

sf_bbox(obj, x = NULL, y = NULL)

Arguments

obj            matrix, data.frame, sfg, sfc or sf object.
x              x geometry column
y              y geometry column

Examples

## data.frame
df <- data.frame(   id1 = c(1,1,1,1,1,1,1,1,2,2,2,2),   id2 = c(1,1,1,1,2,2,2,2,1,1,1,1),   x = c(0,0,1,1,1,1,2,2,3,4,4,3),   y = c(0,1,1,0,1,2,2,1,3,3,4,4) )
sf_bbox( obj = df[, c("x","y")])
sf_bbox( obj = df, x = "x", y = "y")

## sfg objects
pt <- sfg_point(obj = df[1,], x = "x", y = "y", z = "id1")
mpt <- sfg_multipoint(obj = df, x = "x", y = "y")
ls <- sfg_linestring(obj = df, x = "x", y = "y")
mls <- sfg_multilinestring(obj = df, x = "x", y = "y")
p <- sfg_polygon(obj = df, x = "x", y = "y")
mp <- sfg_multipolygon(obj = df, x = "x", y = "y", close = FALSE)

sf_bbox( pt )
sf_bbox( mpt )
sf_bbox( ls )
sf_bbox( mls )
sf_bbox( p )
sf_bbox( mp )

## sfc objects
pt <- sfc_point(obj = df, x = "x", y = "y", z = "id1")
mpt <- sfc_multipoint(obj = df, x = "x", y = "y", multipoint_id = "id1")
ls <- sfc_linestring(obj = df, x = "x", y = "y", linestring_id = "id1")
mls <- sfc_multilinestring(obj = df, x = "x", y = "y", multilinestring_id = "id1")
p <- sfc_polygon(
  obj = df,
  x = "x",
  y = "y",
  polygon_id = "id1",
  linestring_id = "id2",
  close = FALSE
)
mp <- sfc_multipolygon(
  obj = df,
  x = "x",
  y = "y",
  multipolygon_id = "id1",
  linestring_id = "id2",
  close = FALSE
)

sf_bbox( pt )
sf_bbox( mpt )
sf_bbox( ls )
sf_bbox( mls )
sf_bbox( p )
sf_bbox( mp )

## sf objects
pt <- sf_point(obj = df, x = "x", y = "y", z = "id1")
mpt <- sf_multipoint(obj = df, x = "x", y = "y", multipoint_id = "id1")
ls <- sf_linestring(obj = df, x = "x", y = "y", linestring_id = "id1")
mls <- sf_multilinestring(obj = df, x = "x", y = "y", multilinestring_id = "id1")
p <- sf_polygon(
  obj = df,
  x = "x",
  y = "y",
  polygon_id = "id1",
  linestring_id = "id2"
sf_cast

Description

curve the input sf to a different geometry

Usage

sf_cast(sf, to, close = TRUE)

Arguments

sf  object to convert
to  the geometry to convert to.
close  logical indicating if polygons should be closed

Examples

df <- data.frame(
id1 = c(1,1,1,1,1,1,1,1,1,2,2,2,2),
   id2 = c(1,1,1,1,2,2,2,2,1,1,1,1),
   x = c(0,0,1,1,1,1,2,2,3,4,4,3)

mf <- sf_multipolygon(
   obj = df,
   x = "x",
   y = "y",
   multipolygon_id = "id1",
   linestring_id = "id2",
   close = FALSE
)

sf_bbox( pt )
sf_bbox( mpt )
sf_bbox( ls )
sf_bbox( mls )
sf_bbox( p )
sf_bbox( mp )

## you can use it to update a bounding-box if it gets corrupted
attr( mpt, "bbox" ) <- c(1:5)
mpt ## incorrect values
attr( mpt, "bbox" ) <- sf_bbox( mpt )
mpt ## back to correct values
Description

constructs sf of LINESTRING objects

Usage

```r
sf_linestring(
  obj = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  linestring_id = NULL,
  keep = FALSE
)
```
Arguments

- **obj**: sorted matrix or data.frame
- **x**: x geometry column
- **y**: y geometry column
- **z**: z geometry column
- **m**: m geometry column
- **linestring_id**: column of ids for linestrings
- **keep**: logical indicating if the non-geometry and non-id columns should be kept. If TRUE you must supply the geometry and id columns, and only the first row of each geometry is kept. See Keeping Properties.

Value

- **sf** object of LINESTRING geometries

Notes

The `sfheaders` functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

Keeping Properties

Setting `keep = TRUE` will retain the first row of any columns not specified as a coordinate (x, y, z, m) or an id (e.g., linestring_id, polygon_id) of the input obj.

The `sf_*` functions assume the input obj is a long data.frame / matrix, where any properties are repeated down the table for the same geometry.

Examples

```r
x <- matrix( c(1:4), ncol = 2 )
sf_linestring( x )

x <- data.frame( id = 1:2, x = 1:2, y = 2:1 )
sf_linestring( x )
sf_linestring( x, x = "x", y = "y" )
sf_linestring( x, x = "y", y = "x" )
sf_linestring( x, linestring_id = "id", x = "x", y = "y" )
```
Description

constructs an sf of MULTILINESTRING objects

Usage

```r
sf_multilinestring(
  obj = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  multilinestring_id = NULL,
  linestring_id = NULL,
  keep = FALSE
)
```

Arguments

- `obj` sorted matrix or data.frame
- `x` x geometry column
- `y` y geometry column
- `z` z geometry column
- `m` m geometry column
- `multilinestring_id` column of ids for multilinestrings
- `linestring_id` column of ids for linestrings (within multilinestrings)
- `keep` logical indicating if the non-geometry and non-id columns should be kept. if TRUE you must supply the geometry and id columns, and only the first row of each geometry is kept. See Keeping Properties.

Value

sf object of MULTILINESTRING geometries

notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.
Keeping Properties

Setting keep = TRUE will retain the first row of any columns not specified as a coordinate (x, y, z, m) or an id (e.g., linestring_id, polygon_id) of the input obj.

The sf_ functions assume the input obj is a long data.frame / matrix, where any properties are repeated down the table for the same geometry.

Examples

```r
m <- matrix(c(0,0,0,1,1), ncol = 3)
sf_multilinestring( m )

m <- matrix(c(0,0,0,0,1,1,2,2,1,2,3), ncol = 3, byrow = TRUE)
sf_multilinestring( obj = m )
sf_multilinestring( obj = m, multilinestring_id = 1 )
sf_multilinestring( obj = m, linestring_id = 1 )
sf_multilinestring( obj = m, linestring_id = 1, multilinestring_id = 1 )
sf_multilinestring( obj = m, x = 2, y = 3 )
sf_multilinestring( obj = m, x = 1, y = 2, z = 3 )
sf_multilinestring( obj = m, x = 2, y = 3, linestring_id = 1, multilinestring_id = 1 )
df <- data.frame(
  ml_id = c(1,1,1,1,1,1,2,2,2,2,2)
  , l_id = c(1,1,1,2,2,3,3,3,1,1,2,2)
  , x = rnorm(13)
  , y = rnorm(13)
  , z = rnorm(13)
  , m = rnorm(13)
)
sf_multilinestring( obj = df, x = "x", y = "y")
sf_multilinestring( obj = df, x = "x", y = "y", z = "z")
sf_multilinestring( obj = df, x = "x", y = "y", z = "z", m = "m")
sf_multilinestring( obj = df, x = 3, y = 4)
sf_multilinestring( obj = df, x = 3, y = 4, z = 5)
sf_multilinestring( obj = df, x = 3, y = 4, z = 5, m = 6 )
sf_multilinestring( obj = df, multilinestring_id = "ml_id", linestring_id = "l_id" )
sf_multilinestring( obj = df, multilinestring_id = 1, linestring_id = 2 )
```
Description

constructs sf of MULTIPOINT objects

Usage

sf_multipoint(
  obj,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  multipoint_id = NULL,
  keep = FALSE
)

Arguments

obj sorted matrix or data.frame
x x geometry column
y y geometry column
z z geometry column
m m geometry column
multipoint_id column of ids for multipoints
keep logical indicating if the non-geometry and non-id columns should be kept. if TRUE you must supply the geometry and id columns, and only the first row of each geometry is kept. See Keeping Properties.

Value

sf object of MULTIPOINT geometries

notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

Keeping Properties

Setting keep = TRUE will retain the first row of any columns not specified as a coordinate (x, y, z, m) or an id (e.g., linestring_id, polygon_id) of the input obj.

The sf_* functions assume the input obj is a long data.frame / matrix, where any properties are repeated down the table for the same geometry.
Examples

```r
x <- matrix( c(1:4), ncol = 2 )
sf_multipoint( x )

x <- data.frame( id = 1:2, x = 1:2, y = 2:1 )
sf_multipoint( x )
sf_multipoint( x, x = "x", y = "y" )
sf_multipoint( x, x = "y", y = "x" )
sf_multipoint( x, multipoint_id = "id", x = "x", y = "y" )
```

**Description**

constructs an sf of MULTIPOLYGON objects

**Usage**

```r
sf_multipolygon(
  obj = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  multipolygon_id = NULL,
  polygon_id = NULL,
  linestring_id = NULL,
  close = TRUE,
  keep = FALSE
)
```

**Arguments**

- `obj` : sorted matrix or data.frame
- `x` : x geometry column
- `y` : y geometry column
- `z` : z geometry column
- `m` : m geometry column
- `multipolygon_id` : column of ids for multipolygons
- `polygon_id` : column of ids for polygons
- `linestring_id` : column of ids for lines (within polygons)
**sf_multipolygon**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>logical indicating whether polygons should be closed. If TRUE, all polygons will be checked and force closed if possible</td>
</tr>
<tr>
<td>keep</td>
<td>logical indicating if the non-geometry and non-id columns should be kept. If TRUE you must supply the geometry and id columns, and only the first row of each geometry is kept. See Keeping Properties.</td>
</tr>
</tbody>
</table>

**Value**

_sf_ object of MULTIPOLYGON geometries

**notes**

_sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes. The data.frame and matrices you send into the sfheader functions must be ordered.

**Keeping Properties**

Setting keep = TRUE will retain the first row of any columns not specified as a coordinate (x, y, z, m) or an id (e.g., linestring_id, polygon_id) of the input obj.

The _sf_* functions assume the input obj is a long data.frame / matrix, where any properties are repeated down the table for the same geometry.

**Examples**

```r
m <- matrix(c(0,0,0,0,1,0,1,0,1,0,1,0,0,0), ncol = 3, byrow = TRUE )
sf_multipolygon( m )

df <- data.frame(
  id = c(1,1,1,1),
  x = c(0,0,1,1),
  y = c(0,1,1,0)
)
sf_multipolygon( df, x = "x", y = "y" )

df <- data.frame(
  id = c(1,1,1,1,2,2,2,2),
  x = c(0,0,1,0,1,2,1,2),
  y = c(0,1,0,0,1,2,1,1)
)
sf_multipolygon( df, multipolygon_id = "id", polygon_id = "id", linestring_id = "id")

df <- data.frame(
  id1 = c(1,1,1,1,1,1,1,1),
  id2 = c(1,1,1,1,2,2,2,2),
  x = c(0,0,1,1,0,1,2,2),
  y = c(0,1,0,0,1,2,1,1)
)
```
sf_multipolygon( df, multipolygon_id = "id1", polygon_id = "id2" )

df <- data.frame(
  id1 = c(1,1,1,1,1,1,1,1,2,2,2,2),
  id2 = c(1,1,1,1,1,2,2,2,2,1,1,1),
  x = c(0,1,0,1,2,2,1,3,3,4,4,3),
  y = c(0,1,0,1,2,2,1,3,4,3,4,3)
)

sf_multipolygon( df, multipolygon_id = "id1", polygon_id = "id2" )

df <- data.frame(
  id1 = c(1,1,1,1,2,2,2,2),
  id2 = c(1,1,1,1,1,1,1,1),
  x = c(0,0,1,0,1,2,2,1),
  y = c(0,1,0,1,2,2,1,1)
)

sf_multipolygon( df, multipolygon_id = "id1", polygon_id = "id2" )
sf_multipolygon( df, polygon_id = "id1", linestring_id = "id2" )
sf_multipolygon( df, x = "x", y = "y", polygon_id = "id1" )
sf_multipolygon( df, x = "x", y = "y", polygon_id = "id1", linestring_id = "id2" )
sf_multipolygon( df, x = "x", y = "y", linestring_id = "id1" )
sf_multipolygon( df, x = "x", y = "y", linestring_id = "id2" )

df <- data.frame(
  id1 = c("a","a","a","a","b","b","b","b"),
  id2 = c(1,1,1,1,1,1,1,1),
  x = c(0,0,1,0,1,2,2,1),
  y = c(0,1,0,1,2,2,1,1)
)

sf_multipolygon( df, x = "x", y = "y", polygon_id = "id1" )

---

**sf_point**

**sf POINT**

**Description**

constructs sf of POINT objects

**Usage**

sf_point(obj, x = NULL, y = NULL, z = NULL, m = NULL, keep = FALSE)
Arguments

- **obj**: sorted vector, matrix or data.frame
- **x**: x geometry column
- **y**: y geometry column
- **z**: z geometry column
- **m**: m geometry column
- **keep**: logical indicating if the non-geometry and non-id columns should be kept. If TRUE you must supply the geometry and id columns, and only the first row of each geometry is kept. See Keeping Properties.

Value

- **sf**: object of POINT geometries

Keeping Properties

Setting `keep = TRUE` will retain the first row of any columns not specified as a coordinate (x, y, z, m) or an id (e.g., linestring_id, polygon_id) of the input `obj`.

The sf_* functions assume the input `obj` is a long data.frame / matrix, where any properties are repeated down the table for the same geometry.

Notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.

Examples

```r
x <- c(1:3)
sf_point( x )

x <- matrix( c(1:10), ncol = 2 )
sf_point( x )

x <- setNames( as.data.frame( x ), c("x","y") )
sf_point( x )
sf_point( obj = x, x = "x", y = "y" )
sf_point( obj = x, x = "y", y = "x" )

# keeping properties
x$val <- letters[1:5]
sf_point( x, x = "x", y = "y", keep = TRUE )
```
Description

constructs an sf of POLYGON objects

Usage

```r
sf_polygon(
  obj = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  m = NULL,
  polygon_id = NULL,
  linestring_id = NULL,
  close = TRUE,
  keep = FALSE
)
```

Arguments

- `obj` sorted matrix or data.frame
- `x` x geometry column
- `y` y geometry column
- `z` z geometry column
- `m` m geometry column
- `polygon_id` column of ids for polygons
- `linestring_id` column of ids for lines (within polygons)
- `close` logical indicating whether polygons should be closed. If TRUE, all polygons will be checked and force closed if possible
- `keep` logical indicating if the non-geometry and non-id columns should be kept. if TRUE you must supply the geometry and id columns, and only the first row of each geometry is kept. See Keeping Properties.

Value

sf object of POLYGON geometries

Notes

sfheaders functions do not perform any validity checks on the geometries. Nor do they set Coordinate Reference Systems, EPSG, PROJ4 or precision attributes.

The data.frame and matrices you send into the sfheader functions must be ordered.
Keeping Properties

Setting keep = TRUE will retain the first row of any columns not specified as a coordinate (x, y, z, m) or an id (e.g., linestring_id, polygon_id) of the input obj.

The sf_* functions assume the input obj is a long data.frame / matrix, where any properties are repeated down the table for the same geometry.

Examples

```r
m <- matrix(c(0,0,0,0,1,1), ncol = 2 )
sf_polygon( m )

m <- matrix(c(0,0,0,0,0,1,1,2,2,2,1,1,2,1,2,2,1,3,1,3,4), ncol = 3, byrow = TRUE)
sf_polygon( obj = m )
sf_polygon( obj = m, polygon_id = 1 )
sf_polygon( obj = m, linestring_id = 1 )
sf_polygon( obj = m, linestring_id = 1, polygon_id = 1 )
sf_polygon( obj = m, x = 2, y = 3 )
sf_polygon( obj = m, x = 1, y = 2, z = 3 )
sf_polygon( obj = m, x = 2, y = 3, linestring_id = 1, polygon_id = 1 )

df <- data.frame(  
  ml_id = c(1,1,1,1,1,1,1,1,1,2,2,2,2,2,2),  
  l_id = c(1,1,1,2,2,2,3,3,3,1,1,1,2,2,2),  
  x = rnorm(15),  
  y = rnorm(15),  
  z = rnorm(15),  
  m = rnorm(15)  
)
sf_polygon( obj = df, x = "x", y = "y")
sf_polygon( obj = df, x = "x", y = "y", z = "z")
sf_polygon( obj = df, x = "x", y = "y", z = "z", m = "m")
sf_polygon( obj = df, x = 2, y = 3)
sf_polygon( obj = df, x = 2, y = 3, z = 4)
sf_polygon( obj = df, x = 2, y = 3, z = 4, m = 5)
sf_polygon( obj = df, polygon_id = "ml_id", linestring_id = "l_id" )
sf_polygon( obj = df, polygon_id = 1, linestring_id = 2 )
```

---

**sf_remove_holes**

**remove holes**
Description

Removes holes from polygons and multipolygons. Points and linestrings are unaffected.

Usage

sf_remove_holes(obj, close = TRUE)

Arguments

obj sfg, sfc or sf object.
close logical indicating whether polygons should be closed. If TRUE, all polygons will be checked and forced closed if possible.

Examples

df <- data.frame(
  ml_id = c(1,1,1,1,1,1,1,1,1,2,2,2,2,2,2),
  l_id = c(1,1,1,2,2,2,3,3,3,1,1,1,2,2,2),
  x = rnorm(15),
  y = rnorm(15),
  z = rnorm(15),
  m = rnorm(15)
)
sfg <- sfg_polygon( obj = df, x = "x", y = "y", linestring_id = "ml_id" )
sfc <- sfc_polygon( obj = df, x = "x", y = "y", polygon_id = "ml_id", linestring_id = "l_id" )
sf <- sf_polygon( obj = df, x = "x", y = "y", polygon_id = "ml_id", linestring_id = "l_id" )

sf_remove_holes( sfg )
sf_remove_holes( sfc )
sf_remove_holes( sf )

sf_to_df

Description

Converts an sf object to a data.frame

Usage

sf_to_df(sf, fill = FALSE)

Arguments

sf sf object
fill logical indicating if the resulting data.frame should be filled with the data columns from the sf object. If TRUE, each row of data will be replicated for every coordinate in every geometry.
Examples

df <- data.frame(
  ml_id = c(1,1,1,1,1,1,1,1,1,2,2,2,2,2,2),
  l_id = c(1,1,1,2,2,2,3,3,3,1,1,1,2,2,2),
  x = rnorm(15),
  y = rnorm(15),
  z = rnorm(15),
  m = rnorm(15)
)

sf <- sf_polygon( obj = df, polygon_id = "ml_id", linestring_id = "l_id" )
df <- sf_to_df( sf )

## with associated ata
sf$val1 <- c("a","b")
sf$val2 <- c(1L, 2L)

df <- sf_to_df( sf, fill = TRUE )
Index

sf_bbox, 20
sf_cast, 22
sf_linestring, 23
sf_multilinestring, 25
sf_multipoint, 26
sf_multipolygon, 28
sf_point, 30
sf_polygon, 32
sf_remove_holes, 33
sf_to_df, 34
sfc_cast, 2
sfc_linestring, 3
sfc_multilinestring, 4
sfc_multipoint, 6
sfc_multipolygon, 7
sfc_point, 9
sfc_polygon, 10
sfc_to_df, 12
sfg_linestring, 13
sfg_multilinestring, 14
sfg_multipoint, 15
sfg_multipolygon, 16
sfg_point, 17
sfg_polygon, 18
sfg_to_df, 19